HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Design and Engineering 2 (2015) 268–275

www.elsevier.com/locate/jcde

Five-axis finishing tool path generation for a mesh blade based on linear morphing cone

Rong Zhang, Pengcheng Hu, Kai Tang*

Hong Kong University of Science and Technology, Hong Kong

Received 4 March 2015; received in revised form 24 June 2015; accepted 24 June 2015
Available online 2 July 2015

Abstract

Blisk is an essential component in aero engines. To maintain good aero-dynamic performance, one critical machining requirement for blades on blisk is that the generated five-axis tool path should be boundary-conformed. For a blade discretely modeled as a point cloud or mesh, most existing popular tool path generation methods are unable to meet this requirement. To address this issue, a novel five-axis tool path generation method for a discretized blade on blisk is presented in this paper. An idea called Linear Morphing Cone (LMC) is first proposed, which sets the boundary of the blade as the constraint. Based on this LMC, a CC curve generation and expansion method is then proposed with the specified machining accuracy upheld. Using the proposed tool path generation method, experiments on discretized blades are carried out, whose results show that the generated tool paths are both uniform and boundary-conformed.

Keywords: Five-axis tool path; Mesh; Blisk machining; Boundary-conformed; Linear morphing cone

1. Introduction

Blisk is an important component in aero engines. Because of their complex shapes, blades on a blisk are exclusively manufactured by five-axis machining for its unique advantage of large range of machinability and good machining accuracy.

Nowadays, tool path planning plays an important role with regard to computer-aided design, concurrent engineering and their related topics [1]. There are many existing methods for five-axis machining tool path generation, among which the most popular ones are the iso-parametric [2,3], the iso-planar [4–6] and the iso-cusp height [7–11] method. For these three methods, the tool path is generated based on the criterion of choosing a constant parameter in the machining process, i.e., either a constant u or a constant v in the parametric domain of the blade surface, a set of parallel planes with a constant step-over interval, or a constant cusp height between the neighboring Cutter Contact (CC) curves, respectively. These three methods can offer different CC curve patterns, among which the iso-parametric method is the most popular in blade machining, where the blade must be represented as a pure parametric surface. It is exactly because of this parametric representation, in which the boundary of the blade is naturally an iso-parametric curve, the iso-parametric method is able to satisfy the boundary-conforming requirement. Along with these three most common methods, there are also other machining methods aiming at achieving certain specific objectives in the machining process, e.g., good machining efficiency [12], good dynamic and kinematic behavior of the machine tool [13] and effective cutting conditions for the cutter [14], etc. However, all these methods require that the blade be represented as a pure parametric surface. Since for a blade model obtained via a 3-D scanning or CMM acquisition, it can only be represented by a point cloud or mesh, the above mentioned methods can be hardly used to machine the blade directly.

Towards the tool path generation for a discretized model (hereinafter referred as a mesh, since a point cloud can be easily constructed into a mesh), the most intuitive way is to

*Corresponding author. Tel.: +852 2358 8656.
E-mail addresses: rzhangab@ust.hk (R. Zhang), foxpcheng@gmail.com (P. Hu), mektang@ust.hk (K. Tang).
Peer review under responsibility of Society of CAD/CAM Engineers

http://dx.doi.org/10.1016/j.jcde.2015.06.013

modify the existing methods so as to find their application for a mesh model. With this strategy, the iso-planar method can be applied to generate tool path for a mesh model [6,15,16] because it is straightforward to calculate an iso-planar tool path by intersecting planes with the mesh. The iso-cusp height concept was also incorporated into the mesh model tool path generation [17,18] to shorten the total CC curve length. Unfortunately, similar to that in the parametric case, neither of these two adapted methods for a mesh blade is able to meet the boundary-conforming requirement. To address this conforming issue, Sun and Xu did a series of work [19–23] by re-parameterizing the mesh model into a parametric surface with a domain of a square or circular region, so did Oulee et al. [24]. By choosing iso-parametric curves in the re-parameterized domain, the boundary-conformed tool paths can be generated. The method can also be used to generate tool paths for compound surfaces. However, for this method, two sets of very large linear equations are required to be solved in the re-parameterization process, especially when the mesh is very dense, making this method very computational expensive and also prone to numerical instability. Instead of planning the tool path in the re-parameterized parametric domain, Yang et al. [25] and Li [26] directly carried out the calculation on the surface itself to generate boundary-conformed tool paths; but their methods are very complicated and difficult to implement.

To address the above mentioned issues, we in this paper propose a new method of 5-axis tool path generation for a blade represented by a mesh. This method is based on a concept called Linear Morphing Cone (LMC), which is defined according to the geometric properties of the blade on blisk. With the help of LMC, CC curves are constructed by intersecting the mesh model with the LMC. For a blisk with a cone-shaped hub, the proposed method can fulfill the boundary-conforming requirement while the specified machining accuracy (i.e., the cusp height) is upheld. The concept can also be easily extended to cases when the hub is represented by not a cone but a general revolution surface.

In the following part of the paper, the details of the algorithm of constructing the LMC will be introduced first, in Section 2; and then the tool path generation algorithm based on the constructed LMC will be presented, in Section 3; after that, experiments on two blade models will be described in Section 4; and finally we conclude the paper in Section 5.

2. Preliminary

2.1. Geometric property of blade on blisk

A blisk is usually composed of a hub with dozens of evenly distributed blades (sometime with splitters) mounted round it. Fig. 1 shows an example (of part) of a blisk with the hub and three blades. For the hub as shown in Fig. 1 (also the bottom surface of any blade), it is a surface of revolution and usually a lateral surface of a cone or cylinder. Also, for the top surface of a blade as shown in Fig. 1, it is always a surface trimmed from a lateral surface of another cone or cylinder. Thus, the two

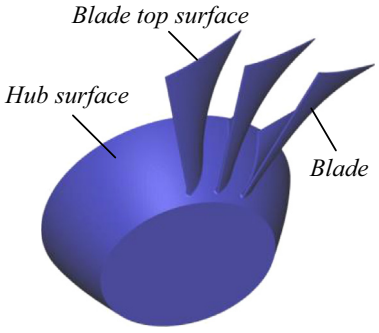

Fig. 1. Elements of a blisk.

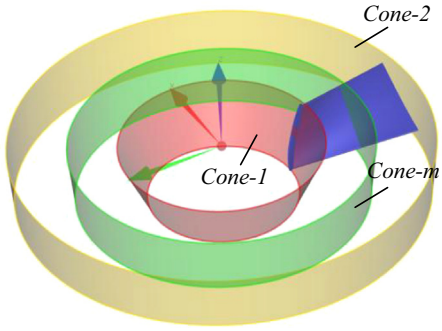

Fig. 2. Cones defined on a blade.

boundaries of the blade on blisk, as shown in Fig. 2, lie on two cones.

Note that cylinder and cone are similar to each other: both of their lateral surfaces are generated by revolving a line (also called generatrix line) around an axis for 2π degrees. Without losing any generality, we use *Cone*-1 and *Cone*-2 to denote the two revolted surface, as shown in Fig. 2.

Clearly, for a blade on blisk, *Cone*-1 and *Cone*-2 are defined according to its boundaries. The generated CC curves which are cone-conformed also conform to the two boundaries of the blade. For a cone-conforming (also boundary-conforming for a blade on blisk) tool path, the first and last CC curve should lie on *Cone*-1 and *Cone*-2, respective, while the rest of CC curves should be uniformly distributed between those two cones. Motivated by this simple fact, we can utilize the two cones, i.e., *Cone*-1 and *Cone*-2, to first generate a set of cones that linearly morph from one to the other, and then intersect them with the blade surface to generate the desired CC curves.

2.2. Linear morphing cone

For the blade model in Fig. 2, its coordinate system is defined in the center of the bottom plane, as shown in Fig. 3, where e_1 and e_2 are the bottom and top edge of *Cone*-1, respectively. Intersect e_1 and e_2 with the X–O–Z plane results in two points P_1 and P_2. Clearly, the line l_1 passing through P_1 and P_2 is the generatrix of *Cone*-1, as shown in Fig. 3. Similarly, the generatrix line l_2 passing through two points Q_1 and Q_2 can be found for *Cone*-2.

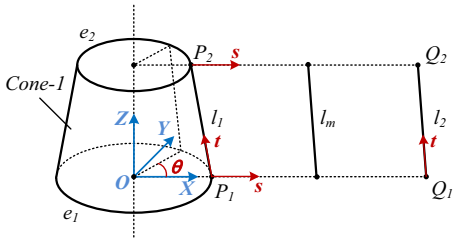

Fig. 3. Definition of the morphing cone.

The generation of the morphing cones consists of two steps. First, the generatrix of the morphing cone, l_m, is calculated by linearly interpolating between l_1 and l_2. Next, we revolve l_m around the Z axis to generate a cone, i.e., the LMC. The detailed derivation steps are described below.

The generatrix line l_1 can be expressed by P_1 and P_2 in a parametric form:

$$l_1 = (1-t)P_1 + tP_2 \tag{1}$$

where $t \in [0, 1]$ is the parameter of this line.

Similarly, the generatrix line l_2 can be expressed as:

$$l_2 = (1-t)Q_1 + tQ_2 \tag{2}$$

where parameter t is the same as that in Eq. (1).

The generatrix line of the morphing cone l_m is a linear interpolation of l_1 and l_2:

$$l_m = (1-s)l_1 + sl_2 \tag{3}$$

where $s \in [0, 1]$ represents the morphing ratio from *Cone*-1 to *Cone*-2.

With the generatrix l_m, the next step is to revolve it around the Z axis to generate the morphing cone *Cone-m*:

$$c_m = \begin{bmatrix} x_m \cos\theta \\ x_m \sin\theta \\ z_m \end{bmatrix} \tag{4}$$

where: x_m, y_m and z_m are the coordinates of l_m; $\theta \in [-\pi, +\pi]$, denoting the revolving angle of l_m around the Z axis.

Eqs. (2) and (3) indicate that *Cone-m* is a linear function of t and s. Eq. (4) indicates that c_m is also a sine function of the third parameter θ, so that it can be expressed as $c_m(t, s, \theta)$. An example of *Cone-m* with $s = 0.5$ is shown in Fig. 2.

With the above derivation, the morphing cone can be constructed as a function of three parameters: t, s and θ, where s is the morphing ratio between *Cone*-1 and *Cone*-2, which determines the shape of the LMC. For a certain s, the other two parameter t and θ define the location of the point on the cone.

With the LMC, the cone-conforming CC curve can be generated by intersecting the LMC with the meshed blade model. Details for this method will be given next.

3. Tool path generation based on LMC

Tool path generation for a meshed blade model based on LMC mainly involves two steps: (1) the CC curve generation, which calculates the CC points so as to obtain a CC curve; and (2) the CC curve expansion, which finds the side step for deciding the next CC curve. Methods for these two steps are explained in details in the next two sub-sections.

3.1. CC curve generation

As already alluded, a CC curve is generated by intersecting the LMC with the meshed blade model. Therefore the CC curve generation is essentially a mesh–cone intersection problem. For a mesh model, its underline component is an edge with two end points. Before solving the mesh–cone intersection problem, the issue of edge–cone intersection should be addressed first.

3.1.1. Edge–cone intersection

Give an edge with two end points T_1 and T_2 as shown in Fig. 4, it can be expressed as:

$$l(r) = (T_2 - T_1)r + T_1 = cr + d \tag{5}$$

where: $c = (T_2 - T_1)$, $r \in [0, 1]$ is the parameter of the edge.

For a particular LMC, s is given, e.g., $s = s_i$, and the LMC can be expressed as $c_m(t, s_i, \theta)$. With l defined in Eq. (5), solving the edge–cone intersection problem, i.e., $l(r)$ intersects with $c_m(t, s_i, \theta)$, resorts to finding the solution of parameters r, θ and t, which can be achieved by solving a 3 by 3 equations:

$$c_m : \begin{bmatrix} (a_1 t + b_1)\cos\theta \\ (a_1 t + b_1)\sin\theta \\ a_3 t + b_3 \end{bmatrix} = l : \begin{bmatrix} c_1 r + d_1 \\ c_2 r + d_2 \\ c_3 r + d_3 \end{bmatrix} \tag{6}$$

where: a_i, b_i, c_i, d_i are the ith component of vectors a, b, c, d that are the parameters constructing the LMC and the edge.

We get:

$$[(a_1 t + b_1)\cos\theta]^2 + [(a_1 t + b_1)\sin\theta]^2 = (c_1 r + d_1)^2 + (c_2 r + d_2)^2 \tag{7}$$

Solve the Z component:

$$t = \frac{c_3}{a_3} r + \frac{d_3 - b_3}{a_3} = m \cdot r + n \tag{8}$$

where: $m = c_3/a_3$, $n = d_3 - b_3/a_3$.

Substitute Eq. (8) to Eq. (7), we get:

$$\left(c_1^2 + c_2^2 - a_1^2 m^2\right) r^2 + 2\left(c_1 d_1 + c_2 d_2 - a_1^2 mn - a_1 m b_1\right) r + \left[d_1^2 + d_2^2 - \left(a_1 n + b_1^2\right)\right] = 0 \tag{9}$$

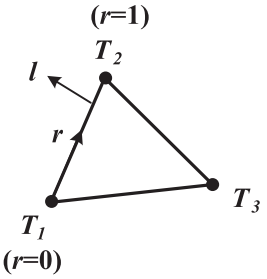

Fig. 4. Definition of mesh structure.

To solve r, the discriminant Δ should be solved. If $\Delta < 0$, there is no intersection between the cone and the line; Else, select the solution of r with $r \in [0, 1]$.

With r solved, the intersection point can be obtained by substituting r into Eq. (5).

3.1.2. Mesh–cone intersection

With the available node–edge–face topology (such as the half-edge data-structure) of the meshed blade obtained in the mesh constructing and storing process, the mesh–cone intersection problem can be easily solved based on the already solved edge–cone intersection problem.

Assume for a triangulated mesh, three lists are constructed to store the nodes, edges and faces of the mesh, denoted as $N = \{n_1, n_2, \ldots, n_i \ldots\}$, $E = \{e_1, e_2, \ldots, e_i \ldots\}$ and $F = \{f_1, f_2, \ldots, f_i \ldots\}$, respectively. The mesh–cone intersection problem can be easily solved with the following three steps.

Step1 Scan the edge list E, until an edge e_m is found intersecting with the cone c_m; this edge is marked as the initial edge intersected by the cone.
Step 2 Based on the edge–face topology, find the left face of e_m, e.g., f_l; and then traverse its three edges, find the other edge e_o (rather than e_m,) which also intersects cone c_m;
Step 3 Substitute e_m with e_o, and repeat **Step 2**, until the very initial edge found in **Step 1** is reached again.

With the node–edge–face topology, the mesh–cone intersection problem can be solved efficiently with the above three steps, with the time-complexity no worse than $O(\log(n))$, where n is the number of edges of the mesh.

3.1.3. Forward step calculation

The intersection between the meshed blade and a particular LMC forms a 3-D closed point loop on the mesh. However, the distribution of those points largely depends on the quality of the mesh model, which may be too dense or too sparse. To guarantee an appropriate machining accuracy, the forward step calculation should be carried out by firstly interpolating the intersection points as a cubic spline and then sampling the CC points according to the specified chord error.

Assume that the interpolated CC curve is $s(t)$, $t \in [0, 1]$, and the specified chord error for the CC point sampling is e, as shown in Fig. 5. For the i-th CC curve p_i with its parameter on $s(t)$ being $t = t_i$, the next CC point p_{i+1} is obtained by calculating the parametric increment Δt_i of the CC curve at p_i, which can be approximated as:

$$\Delta t_i = \sqrt{\frac{8er_i - 4e^2}{I}} \tag{10}$$

where: r_i is the radius of curvature of $s(t)$ at $t = t_i$, I is the length of the tangent vector of $s(t)$ at $t = t_i$, which is $I = ||ds(t)/dt|_{t=t_i}||^2$;

With Δt_i at p_i calculated per Eq. (10), the next CC point p_{i+1} is $s(t)|_{t=t_i+\Delta t_i}$.

In the CC point sampling process, the initial CC point is selected as the point $s(t)|_{t=0}$, and the following CC points are recursively calculated per Eq. (10), until the entire CC curve $s(t)$ is sampled, i.e., when $t = 1$ is reached.

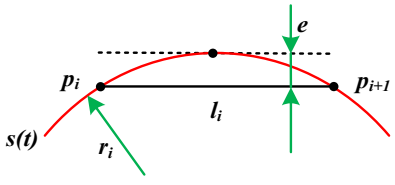

Fig. 5. Forward step on one CC curve.

3.2. CC curve expansion

For each CC curve, CC points are generated by intersecting the LMC with the meshed blade model. Given one CC curve, CC curve expansion is to generate the next CC curve, so that the cusp height between the two neighboring CC curves is bounded by some specified value h. In our particular setting, the CC curve expansion problem is equivalent to: from a given LMC of $s = s_i$, find the next LMC with $s = s_{i+1}$.

Assume that the i-th CC curve is composed of n CC points, e.g., $CC_i = \{CC_{i,0}, CC_{i,1}, \ldots CC_{i,n}\}$, and for the j-th CC point $CC_{i,j}$ on this CC curve, $CC_{i+1,j}$ is the corresponding offsetting point on the next CC curve CC_{i+1}, as shown in Fig. 6.

From the tool path generation scheme proposed in Section 3.1, it is clear that both $CC_{i,j}$ and $CC_{i,j+1}$ are on a certain LMC. Assume that curve CC_i is generated from an LMC of $c_m(t, s_i, \theta)$ with $s = s_i$, and for $CC_{i,j}$, there are corresponding values of t and θ, e.g., $t_{i,j}$ and $\theta_{i,j}$. In this case, $CC_{i+1,j}$ can be expressed by the first order Taylor's expansion at $CC_{i,j}$:

$$\begin{aligned} CC_{i+1,j} = CC_{i,j} &+ \frac{\partial c_m}{\partial t}|_{(s=s_i,\ t=t_{i,j},\theta=\theta_{i,j})} \cdot \Delta t_{i,j} \\ &+ \frac{\partial c_m}{\partial s}|_{(s=s_i,\ t=t_{i,j},\theta=\theta_{i,j})} \cdot \Delta s_{i,j} \\ &+ \frac{\partial c_m}{\partial \theta}|_{(s=s_i,\ t=t_{i,j},\theta=\theta_{i,j})} \cdot \Delta\theta_{i,j} \end{aligned} \tag{11}$$

From another perspective, both CC_i and $CC_{i+1,j}$ belong to the blade surface. To carry out the expansion, a local frame k–f–n should be defined on $CC_{i,j}$, as shown in Fig. 6; where f and n

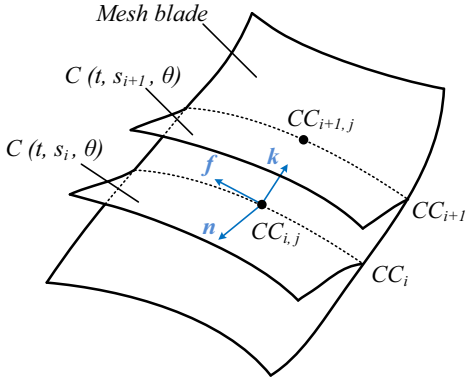

Fig. 6. CC curve expansion.

are the feed direction and surface normal at point $CC_{i,j}$, respectively, and $\vec{k}$ is the cross product of $\vec{f}$ and $\vec{n}$: $\vec{k}=\vec{f}\times\vec{n}$. The expansion distant from $CC_{i,}$ to $CC_{i+1,j}$ are bounded by the machining accuracy requirement.

$$CC_{i+1,j}-CC_{i,j}=\underline{k}\cdot d_{i,j} \tag{12}$$

where $d_{i,j}$ is the cutting strip width at point $CC_{i,}$ and it is decided by the local geometry around point CC_i and the specified maximal cusp height h.

The cutting strip width d can be calculated for a given *cusp* height h as:

$$d=\begin{cases} 2*\sqrt{r^2-\left[\frac{2rR-2hR-h^2}{2(h+R)}\right]^2}/(1-r/R) \text{ (convex case)} \\ 2*\sqrt{r^2-\left[\frac{-2rR+2hR-h^2}{2(h-R)}\right]^2}/(1+r/R) \text{ (concave case)} \end{cases} \tag{13}$$

where: r is the radius of the ball-end cutter, and R is the radius of curvature at the CC point along direction k. Since it is a mesh model, the radius of curvature R for $CC_{i,}$ is estimated with the method proposed in [27].

By combining Eqs. (11)–(13), $\Delta s_{i,j}$ is obtained by solving a 3 by 3 linear equation system. For each CC point $CC_{i,j}$ in $CC_i=\{CC_{i,0}, CC_{i,1},\ldots CC_{i,n}\}$, Δs can be calculated similarly, i.e., $\{\Delta s_{i0}, \Delta s_{i,1}, \ldots \Delta s_{i,n}\}$. To ensure that the cusp height between CC_i and CC_{i+1} not exceed the maximal cusp height h, the minimum Δs among all the CC points on CC_i is selected:

$$\Delta s_i = \min\left(\Delta s_{i,j}\right),\ (j=1,2,\ldots n) \tag{14}$$

With Δs calculated for LMC with $s=s_i$, the next LMC is then readily available:

$$s_{i+1}=s_i+\Delta s_i \tag{15}$$

In our implementation, the first CC curve is generated by intersecting the mesh blade with the LMC of $s=0$ with the method proposed in Section 3.1. From this initial CC curve, expansion is carried out with the equations as described from Eq. (11) to Eq. (15). The two processes – CC curve generation and CC curve expansion – are carried out iteratively, until the whole blade surface is covered Fig. 7.

4. Experimental results and discussion

4.1. Experimental results

To validate the effectiveness of our introduced tool path generation method, experiments are carried out on two meshed blades, as shown in Fig. 8(a) and (b), respectively. In the two experiments, a ball-end cutter with radius smaller than the radius of the blade fillet is chosen, which is 6 mm; the specified cusp height and chord error are both set to be 0.05 mm.

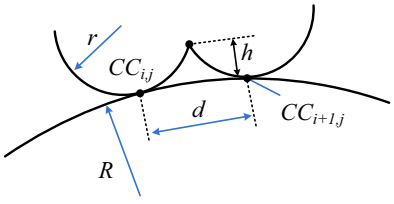

Fig. 7. Cutting strip width.

For blade 1, Fig. 9 is the result of the generated tool path based on LMC. In Fig. 9 (a), *Cone*-1 and *Cone*-2 are defined from the boundaries of the blade, while Cone-*m* is the LMC with coefficient $s=0.5$.

For blade 2, the definition of the machining region is different from that of blade 1: *Cone*-1 is assigned according to the bottom boundary of blade 2 while *Cone*-2 is defined by the user, as shown in Fig. 10(a). Cone-*m* is defined the same way as that for blade-1. The tool path generated for the proposed region is shown in both Fig. 10(a) and (b).

4.2. Discussion

For a meshed blade on blisk, its top and bottom boundary normally lie on two cones; therefore the cone-conformed CC curves also conform to the boundaries of the meshed blade. From the experimental results shown in Figs. 9 and 10, it is clear that the uniform and cone-conformed tool paths generated by the proposed LMC-based method are also boundary-conformed.

There is another advantage with our tool path generation method. Referring to the tool path generation process of blade 2 (Fig. 10): *Cone*-1 is defined according to the bottom of blade 2 while *Cone*-2 is defined by the user. For the region between *Cone*-1 and *Cone*-2, the generated tool path is also uniform and cone-conformed. In this case, neither *Cone*-1 nor *Cone*-2 is required to be fixed on the original model – it can be defined and assigned arbitrarily by the user, meaning that the user has the freedom to select the machining region by selecting proper cones. This property makes our method useful for adaptive machining, in which the blade surface is divided into several regions, and the machining strategy for each region could be different.

Also, with the freedom of selection of proper cones, our tool path generation method can be applied to the machining of other kinds of surfaces, be them parametric or meshed. The only issue here is that the cone-conformed CC curves generated in this way may no longer be boundary-conformed.

With no boundary-conformed requirement, the proposed cone-conformed LMC-based method naturally degenerates into the (generalized) iso-planar tool path generation method. Specifically, if the two bounding cones are coaxial and very large when compared to the workpiece, they could be regarded as two parallel planes from the perspective of the workpiece, and the tool path generated using our algorithm can be regarded as an iso-planar one.

For a tool path generated based on the LMC with ball-end cutter, the generated CC curves are bounded between two cones, e.g., *Cone*-1 and *Cone*-2 as shown in Figs. 9 and 10. In some cases, the containment problem may exist: for the generated tool path, the material removed by the cutter may be outside these two cones, where gouging could happen near the root of the blade. Fig. 11(a) shows an example where CC

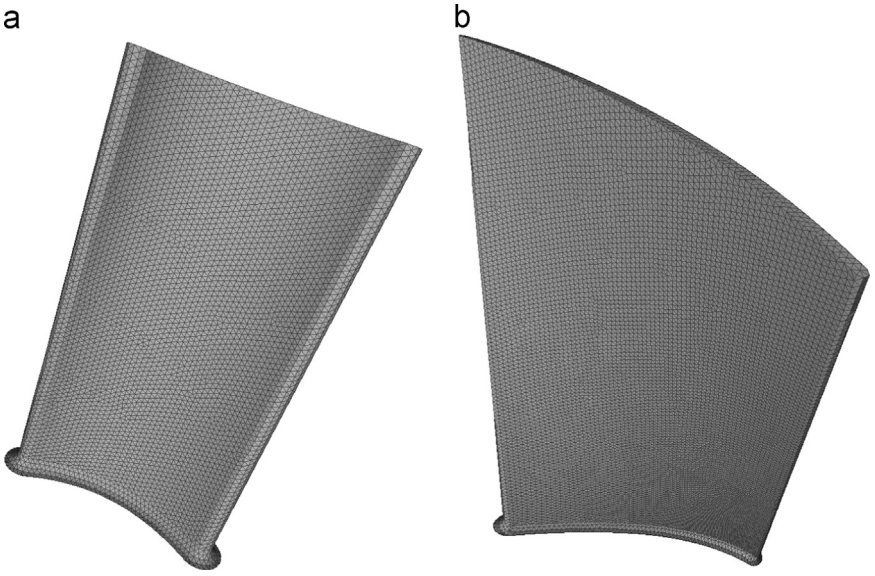

Fig. 8. (a) Meshed blade 1; (b) meshed blade 2.

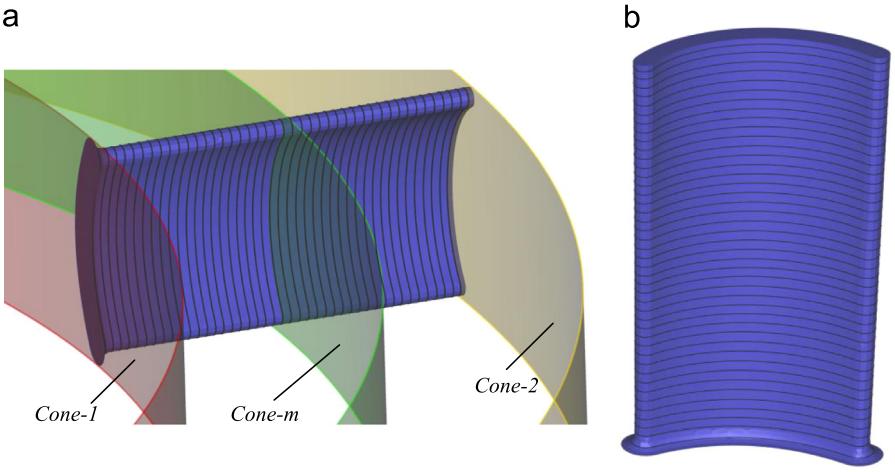

Fig. 9. (a) Cone defining the boundary of meshed blade 1; (b) generated tool path.

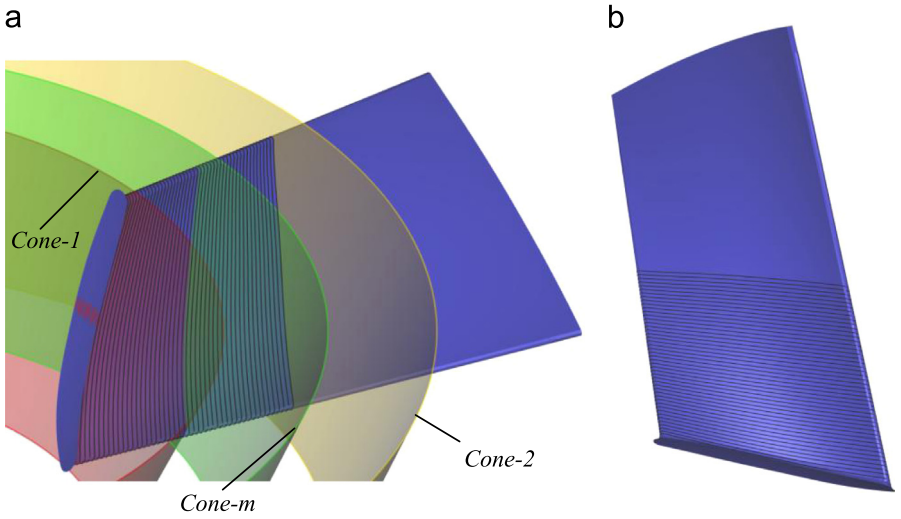

Fig. 10. (a) Cones defining the machining region for meshed blade 2; (b) the generated tool path for the given region.

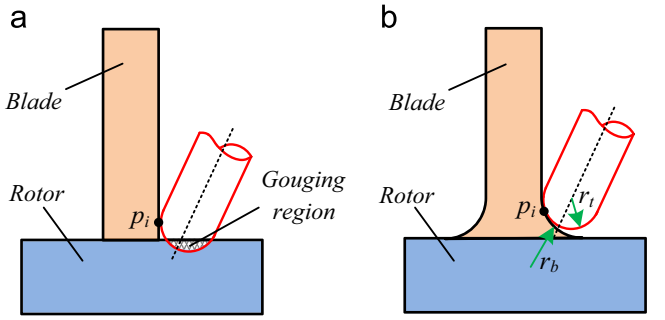

Fig. 11. (a) Case of tool gouging the rotor; (b) case of fillet between the blade and rotor.

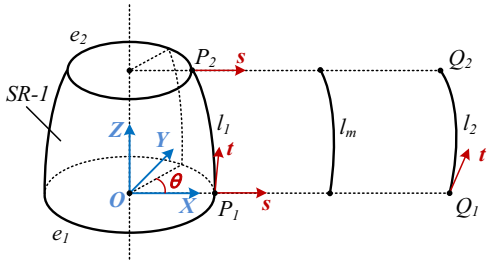

Fig. 12. Definition of the morphing surface of revolution.

point p_i lies on the blade while part of the cutter gouges into the rotor at the same time. Nevertheless, in real machining, this kind of case is very rare. The more general case is that for a blisk, a fillet with radius of r_b is designed to smoothly transit from the blade to the rotor, as shown in Fig. 11(b). In the process of finishing CC curve generation, to guarantee that the entire blade (the blade body and the fillet) can be machined, the radius of the cutter should be smaller than that of the fillet, i.e., $r_t \leq r_b$. In this case, the material removed by the cutter per the generated CC curves is strictly bounded between the two cones, i.e., gouging into the rotor can never happen.

To ensure that the whole fillet surface is covered by CC curves, the first cone, i.e. *Cone*-1, as shown in Figs. 9 and 10, is selected such that it will coincide with the hub surface. Therefore, the intersection curve between the fillet and *Cone*-1, i.e. the first CC curve, is naturally the boundary curve between the fillet and the hub surface. The LMC then propagates from *Cone*-1 towards *Cone*-2 and CC curves can be generated to cover the entire machining region, including the blade surface and the fillet. In this way, the whole fillet and blade surface can be cut thoroughly.

For the proposed CC curve generation method based on LMC, the boundary-conformed property is based on the assumption that the top and bottom boundaries of the blade lie on two corresponding cones, as shown in Fig. 9. As what has been stated in Section 2.1, for the blades on a blisk, this assumption normally holds. However, for other kinds of blades, e.g., blades amounted on a fan or a turbine, things are different because the hub may not be cone-shaped. To extend our method to arbitrary kinds of blades, rather than using the LMC, another type of revolved surfaces called Linear Morphing Revolution Surface (LMRS) can be defined similarly by linearly morphing from the bottom revolved surface SR-1 to the top revolved surface SR-2, as illustrated in Fig. 12. In defining this LMRS, the entire mathematics is the same except that both l_1 and l_2 are now curves instead of lines. Based on LMRS, CC curves can be generated in the same fashion – LMRS–mesh intersection, and they strictly conform to the boundary of the blade.

5. Conclusion

Based on a concept called Linear Morphing Cone (LMC), this paper proposes a five-axis tool path generation method for a mesh model blade on a blisk, which has the unique property of boundary-conforming and at the same time meeting the specified cusp height requirement. This tool path generation method is very useful for meshed blade machining on blisk. By choosing appropriate two cones to define the LMC, the method can be easily applied to five-axis tool path generation for other kinds of part surfaces, e.g., a die or mold, be they parametric or meshed.

Acknowledgment

This work is supported in part by Hong Kong RGC-GRF/16209714, Hong Kong ITS/138/13FX, Hong Kong ITF GHP/057/12, and Hong Kong RGC-GRF/619212.

References

[1] Liu, Y-J, et al. A semantic feature model in concurrent engineering. *Autom. Sci. Eng. IEEE Trans.* 2010;**7**:659–65.
[2] Elber G, Cohen E. Toolpath generation for freeform surface models. *Comput.-Aided Des.* 1994;**26**:490–6.
[3] He, W, et al. Iso-parametric CNC tool path optimization based on adaptive grid generation. *Int. J. Adv. Manuf. Technol.* 2009;**41**:538–48.
[4] Seok Suh Y, Lee K. NC milling tool path generation for arbitrary pockets defined by sculptured surfaces. *Comput.-Aided Des.* 1990;**22**:273–84.
[5] Ding, S, et al. Adaptive iso-planar tool path generation for machining of free-form surfaces. *Comput.-Aided Des.* 2003;**35**:141–53.
[6] Feng H-Y, Teng Z. Iso-planar piecewise linear NC tool path generation from discrete measured data points. *Comput.-Aided Des.* 2005;**37**:55–64.
[7] Suresh K, Yang C. Constant scallop-height machining of free-form surfaces. *J. Eng. Ind.(Trans. ASME)* 1994;**116**:253–9.
[8] Lee E. Contour offset approach to spiral toolpath generation with constant scallop height. *Comput.-Aided Des.* 2003;**35**:511–8.
[9] Can A, Ünüvar A. A novel iso-scallop tool-path generation for efficient five-axis machining of free-form surfaces. *Int. J. Adv. Manuf. Technol.* 2010;**51**:1083–98.
[10] Yoon J-H. Fast tool path generation by the iso-scallop height method for ball-end milling of sculptured surfaces. *Int. J. Prod. Res.* 2005;**43**:4989–98.
[11] Lo C. Efficient cutter-path planning for five-axis surface machining with a flat-end cutter. *Comput.-Aided Des.* 1999;**31**:557–66.
[12] Chiou CJ, Lee YS. A machining potential field approach to tool path generation for multi-axis sculptured surface machining. *Comput.-Aided Des.* 2002;**34**:357–71.
[13] Wang N, Tang K. Five-axis tool path generation for a flat-end tool based on iso-conic partitioning. *Comput.-Aided Des.* 2008;**40**:1067–79.
[14] Xu K, Tang K. Five-axis tool path and feed rate optimization based on the cutting force–area quotient potential field. *Int. J. Adv. Manuf. Technol.* 2014;**75**:1661–79.

[15] Lauwers, B, et al. Development of a five-axis milling tool path generation algorithm based on faceted models. *CIRP Ann.-Manuf. Technol.* 2003;**52**:85–8.
[16] Teng, Z, et al. Generating efficient tool paths from point cloud data via machining area segmentation. *Int. J. Adv. Manuf. Technol.* 2006;**30**: 254–60.
[17] Liu, W, et al. Constant scallop-height tool path generation for three-axis discrete data points machining. *Int. J. Adv. Manuf. Technol.* 2012;**63**: 137–46.
[18] Lee, S-G, et al. Mesh-based tool path generation for constant scallop-height machining. *Int. J. Adv. Manuf. Technol.* 2008;**37**:15–22.
[19] Sun, Y-W, et al. Spiral cutting operation strategy for machining of sculptured surfaces by conformal map approach. *J. Mater. Process. Technol.* 2006;**180**:74–82.
[20] Sun, Y, et al. Contour-parallel offset machining for trimmed surfaces based on conformal mapping with free boundary. *Int. J. Adv. Manuf. Technol.* 2012;**60**:261–71.
[21] Xu J, Jin C. Boundary-conformed machining for trimmed free-form surfaces based on mesh mapping. *Int. J. Comput. Integr. Manuf.* 2013;**26**: 720–30.
[22] Yuwen, S, et al. Iso-parametric tool path generation from triangular meshes for free-form surface machining. *Int. J. Adv. Manuf. Technol.* 2006;**28**:721–6.
[23] Xu, J, et al. Tool path generation by offsetting curves on polyhedral surfaces based on mesh flattening. *Int. J. Adv. Manuf. Technol.* 2013;**64**: 1201–12.
[24] OuLee, T-H, et al. Boundary conformed toolpath generation via Laplace based parametric redistribution method. *J. Manuf. Sci. Eng.* 2004;**126**: 345–54.
[25] Yang, DC, et al. Boundary-conformed toolpath generation for trimmed free-form surfaces. *Comput.-Aided Des.* 2003;**35**:127–39.
[26] Li C. A geometric approach to boundary-conformed toolpath generation. *Comput.-Aided Des.* 2007;**39**:941–52.
[27] M. Meyer, et al., Discrete differential-geometry operators for triangulated 2-manifolds, in Visualization and mathematics III, ed: Springer, 2003, pp. 35–57.